Routing in Wireless Networks with Position Trees

Edgar Chavez, Nathalie Mitton, Hector Tejeda

Routing in Wireless Networks with Position Trees

Edgar Chávez[1], Nathalie Mitton[2], and Héctor Tejeda[1]

[1] Escuela de Ciencias Físico-Matemáticas
Universidad Michoacana de San Nicolás de Hidalgo
Av.Francisco J. Mujica
Morelia - Michoacán - México
[2] IRCICA/LIFL, Univ. Lille 1, CNRS UMR 8022, INRIA Futurs
Parc scientifique de la haute borne - 50, avenue Halley
59650 VILLENEUVE D'ASCQ - France

Abstract. Sensor networks are wireless adhoc networks where all the nodes cooperate for routing messages in the absence of a fixed infrastructure. Non-flooding, guaranteed delivery routing protocols are preferred because sensor networks have limited battery life. Location aware routing protocols are good candidates for sensor network applications, nevertheless they need either an external location service like GPS or Galileo (which are bulky, energy consuming devices) or internal location services providing non-unique virtual coordinates leading to low delivery rates. In this paper we introduce *Position Trees* a collision free, distributed labeling algorithm based on hop counting, which embed a spanning tree of the underlying network . The Routing with Position Trees (RTP) is a guaranteed delivery, non-flooding, efficient implicit routing protocol based on Position Trees. We study experimentally the statistical properties of memory requirements and the routing efficiency of the RPT.
keywords: Location aware routing, virtual coordinates, wireless sensor networks.

1 Introduction

Sensor networks are wireless networks where all the nodes cooperate for routing messages in the absence of a fixed infrastructure. Here the nodes are low cost with limited computational resources and limited battery life. A simple distributed routing algorithm with small memory overhead, and a small CPU demand is thus mandatory for such networks [14]. Typical applications of sensor networks are environment sampling, monitoring disaster areas, security and inventory management.

A route is a sequence of nodes forwarding messages from the source node to the target node. The deployment of new applications in sensor networks heavily relies on the efficiency of route discovery mechanisms. Some (actually deployed) sensor networks use query distribution and data collection based on a model known as data diffusion [15]. In this model, a sink node has a permanent connection with an outside network (*e.g.* the Internet or some wired network) and

performs most of the data analysis, while the other nodes are only used for data acquisition or for simpler data processing. Several protocols have been proposed to express queries over the data sensed by the nodes and to aggregate them [20]. These concepts require support of efficient and robust routing protocols, more powerful than those used to support data diffusion.

Position awareness in sensor networks has improved the efficiency of route discovery and broadcasting algorithms in both power saving and latency measures. The fundamental idea behind position awareness (referred also as *geographic* or *geometric* information) is to provide each node in the network with a label having global information. This information is obtained through devices such GPS or Galileo. Routing protocols based on geographic coordinates of the sensors have revealed to be a very competitive alternative to the classical routing protocols for wireless ad hoc networks, reactive [16, 24] or proactive [8].

Our contribution is a collision-free labeling algorithm based on the uniqueness of a path in a tree. This leads to a non flooding, guaranteed delivery routing protocol (as long as the nodes in the path does not vanish while the packet is in transit). Our routing algorithm presents the same good properties as classical geographical routing, yet it outperforms them, as shown with thorough experimentation, and they don't rely on external location services.

The rest of this paper is organized as follows. Section 2 reports geographic routing over virtual coordinates in literature. Section 3 describes our contribution: the *Routing with Position trees* (RPT). Simulation results are reported in 4 and Section 5 sketches some conclusions and future work.

2 Related Work

When nodes are aware of their geographic coordinates (for instance by the use of a GPS), a geographic routing based on these coordinates is feasible. The greedy approach is called *Most Forward Routing (MFR)* [28]. In MFR, the source node forwards the message to the node that is closest to the destination. This is a simple localized algorithm that does not guarantee delivery. A package can be trapped in a local minimum and the algorithm fails to find a path to the destination leading to low delivery rates. In dense networks the algorithm performs well. Other geographic approaches have then been proposed in order to minimize the energy consumption [17] or to guarantee delivery [12, 18]. The face routing algorithm [12] is generally used to overcome the local optima trap problem of the greedy approach *MFR*. This is performed by extracting a planar subgraph of the network and forwarding the package through the faces. This geographic approaches assume that all nodes are equipped with a GPS, which can become energy-costly and expensive. Moreover, GPS work only in out-door environments.

In order to reduce the cost, one can also equip only a subset of the nodes with a GPS and use these *special nodes* like in [9, 5] to infer the position of the remaining nodes. In such case it suffices to know the distance relative to the *special nodes* using techniques such as time difference of arrival [25], angle

of arrival [23], or signal strength [22]. Once the position of every node in the network in estimated one can use, in theory, any geographic routing algorithm. These approaches introduce, however, drawbacks to the geographic routing. It is possible, for example, that two nodes obtain the same coordinates leading to delivery failures.

Other approaches rely exclusively on the relative distances (or hop counting) to a set of nodes in the network, without the intervention of external location services. The general idea is to define a virtual coordinate system and use it to induce a routing protocol based on the virtual coordinates. We survey some of them below.

The authors of [6] introduce VCap (*Virtual Coordinate assignment protocol*) to support geographic routing. A system of virtual coordinates based on three *landmarks* is proposed. Nodes are assigned a triplet of coordinates given as the number of hops the node is distant from each *landmark*. A more accurate coordinate system can be established as the number of *landmarks* increases [2]. Then, nodes use a greedy routing, like in *MFR*, based on the Hamming distance computed on these coordinates (instead of the Euclidean distance in the original MFR). The storage overhead for each sensor is limited to the storage of its coordinates and the coordinates of its neighbors. If the routing reaches a node v with a local minimum, they give a *local detour* rule which consist in locally flooding within a finite neighborhood. A similar idea is presented in [11] with different tie break and recovery mechanisms. Another approach also based in hop counting is presented in [26] where loops are avoided recording the moving history, a time to live for dropping packages and elaborate tie break and recovery mechanisms. A different approach is used in [10] where landmarks are selected more carefully after partitioning the nodes into tiles, and elaborate gradient descent procedures are used to route packets, and high communication and storage overhead is required to increase the delivery rate.

All these geographic routings relying on virtual coordinates use landmarks and the number of hops each node is distant from each *landmark* to compute node coordinates. None of them guarantees delivery, unless significantly increasing the resource consumption (e.g. by flooding the network). The core problem of such routing is the amount of reference points needed to produce a unique reference framework. With hop counting, the labels or identifiers obtained are not unique while the routing algorithms rely on the uniqueness of the labels. The authors of [19] studies the number of these reference points needed to avoid duplicate labels. The main result states that the number of beacons needed to get the set fixed is linear on the number of nodes, hence a logical coordinate system must use $O(n)$ such reference points, and consequently must use $O(n)$ space to store the labels. Our result is in line with the bounds given in [19] for nodes in general position. The key difference is that we encode the labels in a different way, using only $O(\log n)$ space to represent each label.

In this paper, we propose a geographic routing algorithm based on virtual coordinates.

We use the uniqueness of a path in a tree to assign labels to nodes and assure the labels uniqueness. As far as we know, this approach using this tree feature has only been used in [21] but it can be used only over a clustering structure.

3 Routing with Position Trees (RPT)

3.1 Position Tree Routing Algorithm

Basic Idea In this paper, we propose a geographic routing algorithm based on virtual coordinates. The objective of a logical coordinate system is to fix all the nodes in a network making it invariant under rigid transformations. This is a theoretical requirement motivated by the observation that a dynamic routing algorithm must handle the packet to the next hop using only local information. If two nodes share the same description, then a local and deterministic decision can lead to only one route (unless the packet is divided, i.e. flooding the network), hence one of the nodes will not receive the packet. In a geographic routing using an external positioning service for each node, the network is fixed in the sense above. Our goal is then to fix the network in the sense above by using an **internal** positioning service while requesting the smallest amount of memory at each node.

Firstly we select and arbitrary root node and label hierarchically from root to child nodes (child nodes are all the unlabeled neighbors of the current root). Child nodes inherit their parent's label plus an extra number. Hence, the path from the root to each node is encoded in the label. A node's label is built from left to right, and it describes the lineage of the node. The size of the label is proportional to the length of the path from the nodes to the root, and is bounded by the height of the tree ($O(\log n)$ on the average if the tree is balanced).

The labeling is hierarchical. The root node arbitrarily enumerates its neighbors, it can even *skip a neighbor* (following an arbitrary enumeration heuristic optimizing an external goal). As a result of this observation, we can build a virtual topology that can be unrelated to the physical topology of the network, optimizing external goals (e.g. the reliability of the nodes), and the correctness of the algorithm will not be compromised.

Labels are thus used to perform an interval routing over the tree. Interval Routing was introduced in wired networks by Santoro and Khatib in [27] to reduce the size of the routing tables. It is based on representing the routing table stored at each node in a compact manner, by grouping the set of destination addresses that use the same output port into intervals of consecutive addresses. The main advantage of this scheme is the low memory requirements to store the routing on each node u: $O(\delta(u))$. The routing is computed in a distributed fashion with the following algorithm: at each intermediate node x, the routing process ends if the destination y corresponds to x, otherwise, it is forwarded with the message through an edge labeled by a set I such that $y \in I$. The algorithm is shown to be optimal for acyclic graphs, and it exhibits a worst-case complexity which is a factor of two from the optimal solution for an arbitrary topology.

For naming nodes, authors of [27] construct a minimum-distance spanning tree and traverse in depth-first style assigning a distinct integer to each node. In our algorithm, this step is not necessary. For assigning labels to the neighbors, they use the spanning tree, in our case we use all the neighbors that are enabled, and this step is done locally. They gave some limitations when nodes are added o deleted, and with permanent disconnections a new tree must be constructed.

Finally, since we use only the neighborhood relationship both when labeling the nodes and routing packages, without additional hypothesis, it is possible to route in any arbitrary network, in particular in three dimensions.

Assumptions We consider a sensor network composed of nodes uniformly scattered. The nodes are assumed static, or with a very low mobility with respect to the signal propagation speed. We assume that every node has a unique ID.

Once each node is labeled, the routing task can be run. To send a message to a destination node u, a node v needs to know the label of node u. It thus need a locating protocol such that [3, 21] which retrieves a node label from its identity. We do not consider this locating part in this paper and assume that there exists such a locating service available in the network. Note that this service is required for any routing protocol.

Labeling the nodes The labeling process is very simple. First, a node of the network is chosen to be the *"root"*. It is labeled as the root R. The root node can be selected randomly, or with some heuristic. The selection of the root is not central to the correctness of our algorithm, although different selections will produce different dilations in a given path.

The root R advertises all its neighbors that they are the root's children. For it, it broadcasts a "Discovery" request. Each neighbor of R answers the root node by sending a "Tag Request" message containing its ID. Since this is the root which instances the labeling process, none of its children has already been labeled. The root sorts the ID of its children and is then able to assign a label to each of them. Labels are of the form Rm where m is a positive integer chosen by R, according to the children sort of R.

Once a node u has received its label, it is ready to name its own children in a similar fashion. Node u broadcasts a "Discovery" request to its neighbors. Only the ones of them which have not been labeled by another node answer with a "Tag Request" message. Parent node u sorts the ID of its non-tagged neighbors which become its children in the labeled tree. Node u assigns them a label. If the parent's label is a string s, its children's labels will be obtained by appending a positive integer to s. Since in wireless environments, a transmission by each node reaches all nodes within radius distance from it, node u can send the labels of all its children in only one message instead of sending one message per neighbor node.

This process is iterated until all the nodes in the network have been labeled. Once this happens, we must tell the root the labeling process is done. When a node becomes "childless", that is, all its neighbors are either been labeled or

tagged by other nodes, it sends its parent a message informing it that its descendants have been completely determined. When a parent node has received similar messages from all its children, (*e.g* all descendants have been determined), it advertises its own parent, until the root gets the message from all its children. At that time, all the nodes of the network have been labeled.

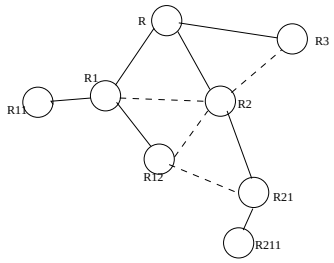


Fig. 1: Labeling process. Dark links are links of the tree, dashed links are the wireless underling links.

Figure 1 illustrates the labeling process over an arbitrary topology. Dark lines represent the links in the tree. Dashed lines are the wireless links which have not been selected to be part of the tree during the labeling process.

Storage requirements Once every node is tagged, the routes are computed and the network is ready for routing. The nodes need to store just their own label. As we will see below, nodes do not even need to store their neighbor's labels.

Sending a Packet Once the labels are in place, sending a packet through the network is almost trivial, because the labels define a canonical a path joining any pair of nodes.

Let's suppose node A wants to send a packet to node B. The packet is forwarded up the tree to find the least common ancestor of A and B and then down the tree until finding node B. All the forwarding decisions are made locally, without flooding. To fix ideas, suppose A is labeled $R167895$ and B is labeled $R16774232$, the largest common prefix is $R167$ which corresponds to the least common ancestor of A and B. By broadcasting the source and destination labels there is only one node, labeled $R16789$, which will forward the packet. This process is repeated with the sequence $A = R167895, R167789, \cdots, R167$ (being the last one the common ancestor) and from there the packet is forwarded to the nodes with unique labels $R1677, R16774, \cdots, R16774232 = B$ to reach node B. The up tree forwarding could be an empty step if the source/destination is the least common ancestor of the destination/source respectively.

The above sketch of the routing protocol is as simple as the actual routing protocol. The actual implementation may rewrite the source in each step. In this case the current node will broadcast the new source, and only the new source node will forward the message rewriting the packet each time.

To illustrate this routing scheme, let's take the tree plotted on Figure 1. Let's suppose node $R211$ wants to send a message to node $R11$. The message will follow the path $R211 - R21 - R2 - R - R1 - R11$. But when node $R21$ forwards the message, nodes $R211$, $R12$ and $R2$ receive it whereas only node $R2$ needs to forward it. In this case R is the least common ancestor between the destination $R211$ and $R11$.

3.2 Analysis

Message complexity The total number of messages needed to label the nodes is linear in both the number of nodes and the total number of connections. To label its children, each node broadcasts a "Discovery" message, contacting at the same time all its neighbors. We thus generate N messages where N is the number of nodes in the network. After all unlabeled children reply to their parent, the parent node sends a unique message for the totality of its children giving them their corresponding labels. We thus generate N_1 messages where N_1 is the number of internal nodes in the labeled tree. To finish the labeling process, each node must inform its parents when all its children are tagged or if it is a leaf in the labeled tree. Hence the total number of messages is $N - 1$ (the root node does not need to send this message). Hence the total number of these messages is thus $2N + N_1 - 1$

Memory complexity Let t be the length of the longest path which starts at the root. The longest possible label has as many entries as t, so the largest possible amount of memory to store a label is t. The length of the path is $t = O(n)$ in worst case and $t = O(\log(n))$ if the tree is balanced. If a balanced tree can be found by e.g. selecting a node in the *center*, then the size of each label will be $O(\log(n))$.

3.3 Enhancements

Improving the routing paths Since there is a single path in the tree from the source to the destination, and since the length of this path can be computed before the actual packet forwarding begins, multiple trees may help to discover more efficient routes. It also provides a simple way to recover from vanishing nodes or congestion problems. A way to improve our routing algorithm is to multiple roots to produce orthogonal labels (a vector of labels) and thus get a robust algorithm.

When running the routing algorithm from node A to B, every nodes' labels are compared and the couple of labels minimizing the path length (*e.g.* the ones which has the greatest common prefix) is chosen.

Nevertheless, maintaining several trees imply more memory requirements and more computations at each node. Yet, there is a trade-off to study between the number of roots and the node resource requirements.

Shrinking the paths Below the logical path defined by the label tree, there could be many physical connections (dashed links in Figure 1), defining a path shorter than the shortest path in the label tree. For example, in Figure 1, if the source node has label $R11$ and the target node has label $R21$, the shortest path in the tree would be $R11 - R1 - R - R2 - R21$. Since nodes $R1$ and $R2$ are neighbors, then we can avoid the upper part of the path.

A simple way to discover this shorter route is by examining the labels. Indeed, by examining the label, a node can compute the length of the path in the tree between itself and the destination. If R is the least common ancestor for nodes $R1$ and $R211$, the length of the path in the tree from $R1$ to $R211$ is equal to the length of the path from R to $R211$ more the length of the path from R to $R21$. Yet, the length of a path from node R and one of its descendant ($R1$ or $R211$) is equal to the difference between the size of the labels. R has size 1. $R1$ has size 2 (R and one integer), $R211$ has size 4 (R and 3 integers), thus hop distance in the tree from $R1$ and $R211$ is 4 $((2-1)+(4-1)$. Hence , a node is able to determine whether the path going through it is smaller than the one following the tree and thus decide whether directly forward the message. This will imply a slight modification of the routing algorithm because the shortcut node need to inform that he will be forwarding the message.

4 Experimental Results

We used our own C simulator assuming an ideal MAC layer, *i.e.* no interferences and no packet collisions. The nodes were randomly deployed in a 1×1 square using a Poisson Point Process (node positions are independent) with different intensity λ (λ represents the mean number of nodes per surface unit). These nodes have the same transmission range, R, therefore, two nodes are connected by an edge if and only if their Euclidean distance is at most R (assuming a Unit Disk Graph [7]). If there exists a link between nodes u and v, we say that nodes u and v are neighbors. We note $\delta(u)$ the degree of node u, *i.e.* the number of neighbors of node u. With such a Poisson node distribution, we have $\lambda\pi R^2 = \tilde{\delta}$. All results obtained are within a 95% - confidence interval.

4.1 Tree Features

First, we are interested in the intrinsic characteristics of the tree the routing is based on. We regard the mean tree depth. Indeed, the mean tree depth gives the latency needed to build the labeled tree. The labels are computed from the tree building and the deeper the tree, the bigger the maximum label size. Since each node only need to store its own label for routing, the maximum label size gives the memory requirements at each node. Tree depth gives a measure for the latency and for the memory requirements. We also computed the average number of messages sent per node for building the tree. This is proportional to the energy cost for building the tree.

First, we run simulations with fixing R to 0.10 with increasing λ. In this way, we can observe the behavior of the tree when the node degree increases while the network diameter remains constant. Figure 2 clearly shows that the parameters are independent of the node degree. This is a consequence of the building process, where each parent node sends only one message for its whole neighborhood, whatever its size. Furthermore, since a node only stores its own label, the memory requirements neither increase. We can thus deduce that when the number of nodes increases in a bounded environment, our labeling protocol scale well since it neither send more messages nor increase the storage needs.

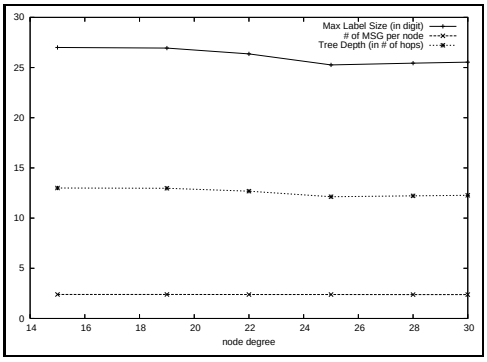
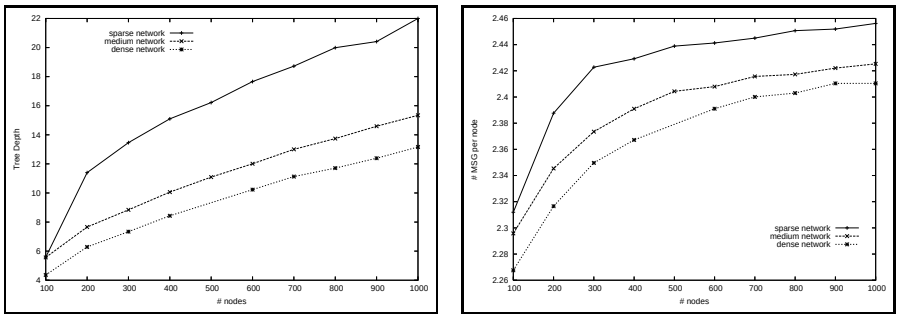


Fig. 2: RPT characteristics for $R = 0$..

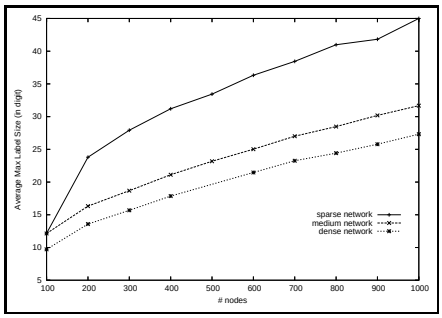
We then consider 2 densities: low (mean node degree 8), and dense (degree 15). For each density, we make λ increase, and consequently the number of nodes increase, as well as the network diameter. Figure 3 shows the results. As expected when the network diameter increases, the tree also growths and thus, more memory is required at each node for storing labels. The number of messages sent remains very low and tends toward a constant asymptote. Simulation results match with the analysis of Section 3.2. The tree depth and the label size are in $O(log(n))$. Please note that we are using a random root node.

4.2 Routing evaluation

We compared our algorithm RPT to other geographic routing protocols. Firstly we compared RPT with MFR [28] and VCap [6]. MFR uses geographic coordinates provided by a satellite receiver like GPS. MFR is thus more expensive than RPT in terms of equipments. Nevertheless, it provides routes with small number of hops and thus gives us a reference point. VCap uses virtual coordinates computed from landmarks nodes. Up to the best of our knowledge, this protocol gives the better results in terms of coordinates computing complexity and success rate among existing routing protocols based on virtual coordinates,



(a) Tree Depth (b) # messages sent per node



(c) Maximum size label

Fig. 3: Characteristics of the tree for $\tilde{\delta} = 8$ and $\tilde{\delta} = 15$.

Table 1 sums up the complexity in term of messages and storage overhead for these algorithms. N is the number of nodes in the network. L refers to the number of landmarks used in the VCap protocol.

	MFR	VCap	RPT
Cost	GPS	$N \times L$ msg	$2N + N_1$ msg
Node memory	own coordinates neighbors' coordinates	own coordinates neighbors' coordinates	own coordinates

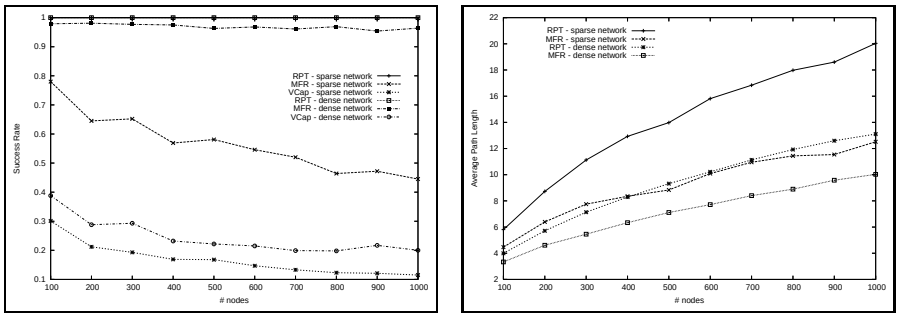
Table 1: Comparative complexity

Note that RPT requires the smaller amount of memory. On can also notice that RPT needs less message exchanges than VCap.

We run the simulation using the routing algorithms for the same samples of node distribution. We considered 2 densities: low (average node degree 8) and dense (average node degree 15). For each density, we make λ increasing, and hence increasing both, the number of nodes increased and the network diameter.

To implement VCap, we use 5 *landmarks* randomly selected from the network nodes.

Figure 4(a) shows the success rate of the protocols studied. As expected, our protocol achieves a 100% of success, and far outperforms the other protocol based on labels (VCap) but also the geographic routing protocol MFR.



(a) Success Rate (b) Average path length (in # of hops)

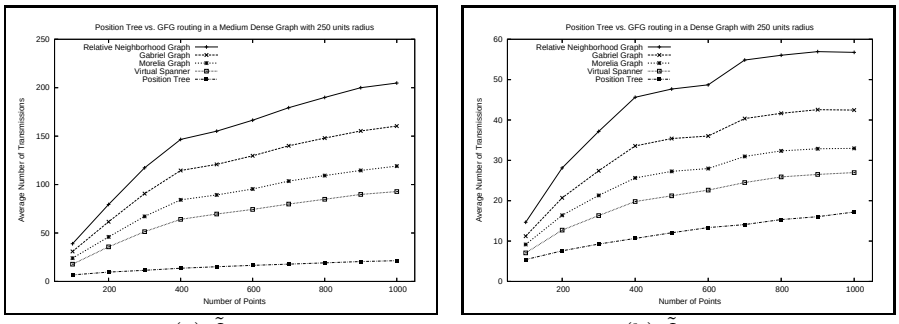
Fig. 4: Routing Characteristics of the tree for $\tilde{\delta} = 8$ and $\tilde{\delta} = 15$.

Since we use a spanning tree of the network, routes are not optimal. We are thus interested in how far the routes provided by RTP are from the optimal. Figure 4(b) plots the average path length of RTP and MFR, MFR providing optimal paths. Since VCAP succeeds only for small paths, its average path length is not significant and we do not consider it in this figure. As expected, the average path length increases as the nodes have less neighbors. When comparing our algorithm vs. shortest paths (MFR), we can observe that the stretch factor is constant, which allows our algorithm to scale with an increasing network diameter.

We guess that the average path length provided by RTP could be greatly improved, particularly in dense networks, with the use of shortcuts as explained in Section 3.3. We keep this experimental analysis for future work.

We also compared RPT with other geographic routing algorithms that guaranteed delivery like Greedy Face Greedy (GFG) [12, 18]. Since the path length depends on the planar subgraph extracted, we compared RPT with RNG [31], Gabriel Graph [13], Morelia Graph [4] and the virtual spanner [30, 29]. Please note that GFG relies on an external location service like GPS. We tested two different network densities for all the algorithms and we show the results in figure 5.

As Figure 5 shows, our RTP protocol outperform the GFG when regarding the length of the providing routes for any planarization method independently of the density.



(a) $\tilde{\delta} = 8$ (b) $\tilde{\delta} = 12$

Fig. 5: Average path length (in # of hops) for $\tilde{\delta} = 8$ and $\tilde{\delta} = 12$ for RTP and GFG using several planarization methods, for $R = 0.25$.

5 Conclusions and Future Work

The label trees defined here are very similar to the coordinates reported in [1], the key difference is the top-down construction of a single tree in our approach. Our contribution also includes the investigation of statistical properties of the label trees. We showed our routing algorithm is very competitive when compared with other routing algorithms. It offers a very good trade-off between equipment cost, number of messages sent, stretch factor and memory requirement. In the extended version of this paper we will report a thorough comparison with other tree-based routing protocols. Below we enumerate some of the trends we may follow for future work:

- Assuming the nodes can adjust the transmission range we may try to build a tree that is least expensive in terms of the energy used for transmitting a package. As it is known that an optimal transmission radius can be found locally, we may choose edges that better approximate this radius.
- Since the underlying structure for routing is a tree, it is natural to explore broadcasting, optimizing e.g. the energy used or the latency in reaching the entire network. In this case a similar technique can be used as in the above case.
- For a fixed tree different root node selections may lead to different label sizes with impact in the amount of memory required for each node. The selection of a core node in a distributed manner is an interesting open problem.
- It will be interesting to study the behavior of RTP in a non-ideal MAC layer.
- The tree maintenance (i.e. taking care of vanishing nodes and the incorporation of new nodes to the network) is a challenging problem, we foresee some strategies based on the analysis of shortcuts in the tree, and the use of multiple roots where we need to balance the memory usage and the failure recovery (this also applies to the case of a non-ideal MAC layer).

We wish to thank the thorough review and suggestions of the anonymous referees who helped to improve the presentation.

References

1. Y. Ben-Asher, M. Feldman, and S. Feldman. Ad-hoc routing using virtual coordinates based on rooted trees. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, pages 6–13, 2006.
2. F. Benbadis, J. Puig, M. D. de Amorim, C. Chaudet, T. Friedman, and D. Simplot-Ryl. JUMPS: Enhanced hop-count positioning in sensor networks using multiple coordinates. *submitted to Elsevier*, 2007.
3. L. Blazevic, S. Giordano, and J. Le Boudec. Anchored path discovery in terminode routing. In *Networking*, Pisa, Italy, 2002.
4. P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J. Opartny, G. Salazar, and J. Urrutia. Morelia test: Improving the efficiency of the gabriel test and face routing in ad-hoc networks. *Lecture Notes in Computer Science*, 3104:23–24, 2004.
5. S. Capkun, M. Hamdi, and J. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *HICSS*, 2001.
6. A. Caruso, S. Chessa, S. De, and A. Urpi. GPS free coordinate assignment and routing in wireless sensor networks. *INFOCOM*, 1:150–160, 2005.
7. B. N. Clark, C. Colbourn, and D. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.
8. T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol (OLSR), October 2003. RFC 3626.
9. E. Ermel, A. Fladenmuller, G. Pujolle, and A. Cotton. On selecting nodes to improve estimated positions. In *MWCM*, 2004.
10. Q. Fang, J. Gao, L. Guibas, V. Silva, and Z. Li. Glider: gradient landmark-based distributed routing for sensor networks. *INFOCOM*, 1:339–350, 2005.
11. R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point in wireless sensornets. Technical Report Tech. Rep. IRB-TR-04-012, Intel Research Berkeley, 2004.
12. H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *MOBICOM*, 2006.
13. K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.
14. J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.
15. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MOBICOM*, 2000.
16. D. Johnson, D. Maltz, and J. Broch. *Ad Hoc Networking, DSR The dynamic source routing protocol for multihop wireless networks*, pages 139–172. 2001.
17. J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress and location based localized power aware routing for ah hoc sensor wireless networks. *IJDSN*, 2:147–159, 2006.
18. J. Li, L. Gewali, H. Selvaraj, and V. Muthukumar. Hybrid greedy/face routing for ad-hoc sensor network. *DSD*, 0:574–578, 2004.
19. P. W. M. Wattenhofer, R. Wattenhofer. Geometric routing without geometry. *Lecture Notes in Computer Science*, 3499:307–322, January 2005.
20. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Operating Systems*, 36:131–146, 2002.
21. N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *AINTEC*, Bangkok, Thailand, December 2005.
22. D. Niculescu and B. Nath. Ad hoc positioning system. In *GLOBECOM*, 2001.

23. D. Niculescu and B. Nath. Ad hoc positioning system (APS) using AOA. In *INFOCOM*, 2003.
24. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-demand Distance Vector Routing, July 2003. RFC 3561.
25. N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *MOBICOM*, pages 1–14, 2001.
26. T. A. Q. Cao. A scalable logical coordinates framework for routing in wireless sensor networks. In *RTSS*, pages 349–358, 2004.
27. N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The computer journal*, 28(1):427–442, 1985.
28. H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE transaction on communications*, com-22(3), 1984.
29. H. Tejeda, E. Chávez, J. Sánchez, and P. Ruiz. Energy-efficient face routing on the virtual spanner. In *ADHOC-NOW*, pages 101–113, 2006.
30. H. Tejeda, E. Chávez, J. Sánchez, and P. Ruiz. A virtual spanner for efficient face routing in multihop wireless networks. In *PWC*, pages 459–470, 2006.
31. G. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1998.